1 SUMMARY

This subroutine **performs an in-place sort on a sparse matrix to an ordering by columns.** There is an option for ordering the entries within each column by increasing row indices and an option for checking for indices that are out of range or duplicated.

ATTRIBUTES — **Version:** 1.0.2. (21 September 2009) **Types:** Real (single, double), Integer, Complex (single, double). **Remark:** This subroutine supersedes `MC20`, `MC39`, `MC49`, `ME20`, and `MF49`. **Calls:** None. **Language:** Fortran 77. **Original date:** December 2000. **Origin:** I.S. Duff and J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 The argument list

There is a single routine that may be called by the user. Note that the user must choose whether or not the matrix data is to be checked for out-of-range indices and for duplicated indices. Performing these checks incurs an overhead. If the user chooses to perform no matrix data checking then, if there are indices that are out of range, the code will abort during execution with no error message and, if there are duplicate entries, these entries will remain in the final order.

The single precision version

```
CALL MC59A(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
```

The double precision version

```
CALL MC59AD(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
```

The integer version

```
CALL MC59AI(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
```

The complex version

```
CALL MC59AC(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
```

The double precision complex version

```
CALL MC59AZ(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
```

`ICNTL` is an `INTEGER` array of length `10` that must be set by the user. It is used to specify control parameters for the subroutine. This argument is unchanged by the routine.

`ICNTL(1)` indicates whether the user-supplied matrix is checked for indices that are out of range and for duplicated indices. If `ICNTL(1)` = `0`, matrix data checking is performed. Otherwise, there is no matrix data checking.

`ICNTL(2)` indicates the ordering requested. The options are:

- `ICNTL(2)` = `0`: input is in arbitrary order and the output is sorted by columns, with the entries within each column in arbitrary order.
- `ICNTL(2)` = `1`: input is in arbitrary order and the output is sorted by columns, with the entries of each column ordered by increasing row indices.
- `ICNTL(2)` = `2`: input is ordered by columns and on output the entries of each column are ordered by increasing row indices.

Restriction: `ICNTL(2)` = `0`, `1`, or `2`.

`ICNTL(3)` indicates whether matrix entries are to be ordered. If `ICNTL(3)` = `0`, matrix entries are ordered.

Otherwise, only the sparsity pattern of the matrix is ordered.

ICNTL(4) is the unit number for error messages. Error messages can be suppressed by setting ICNTL(4) < 0.

ICNTL(5) is the unit number for warning messages. Warning messages can be suppressed by setting ICNTL(5) < 0.

ICNTL(6) indicates the symmetry of the matrix. The options are:
- ICNTL(6) = 0: the matrix is unsymmetric and non Hermitian.
- ICNTL(6) = ±1: the matrix is treated as symmetric and only the lower triangular part of the reordered matrix is returned.
- ICNTL(6) = ±2: the matrix is treated as Hermitian and only the lower triangular part of the reordered matrix is returned.

The sign of ICNTL(6) is important only if data checking is performed (ICNTL(1) = 0) In this case, if ICNTL(6) is positive, the values of duplicate entries are added together and if ICNTL(6) is negative, the value of the first occurrence of the entry is used.
Restriction: ICNTL(6) = 0, ±1, or ±2.

ICNTL(7) to ICNTL(10) are not currently accessed by the routine.

NC is an INTEGER variable that must be set by the user to the number of columns in the matrix. This argument is unchanged by the routine. **Restriction:** NC ≥ 1.

NR is an INTEGER variable that must be set by the user to the number of rows in the matrix. This argument is unchanged by the routine. **Restriction:** NR ≥ 1 and, if ICNTL(6) ≠ 0, NR = NC.

NE is an INTEGER variable that must be set by the user to the number of entries in the matrix. This argument is unchanged by the routine. **Restriction:** NE ≥ 1.

IRN is an INTEGER array of length NE that must be set by the user to hold the row indices of the entries in the matrix as follows:

If ICNTL(2) = 0 or 1, the entries may be in any order. If the matrix is symmetric or Hermitian (ICNTL(6) = ±1 or ±2), for each pair of off-diagonal entries a_{ij} and a_{ji} the user must set the row index of the entry to either *i* or *j*.

If ICNTL(2) = 2, the entries in column J must be in positions IP(J) to IP(J+1)-1 of IRN. If the matrix is symmetric or Hermitian (ICNTL(6) = ±1 or ±2), only entries in the lower triangular part of the matrix should be set (if data checking is used, entries in the upper triangular part are treated as out of range and are removed).

On exit, the row indices are reordered so that the entries of a single column are contiguous with column J preceding column J+1, J = 1, 2, ..., NC-1, with no space between columns. If ICNTL(2) = 0, the order within each column is arbitrary; if ICNTL(2) = 1 or 2, the order within each column is by increasing row indices. If the matrix is symmetric or Hermitian, the lower triangular part of the matrix is returned.

LJCN is an INTEGER variable that must be set by the user to the length of the array JCN. This argument is unchanged by the routine. **Restrictions:** If ICNTL(2) = 0 or 1, LJCN ≥ NE; if ICNTL(2) = 2, LJCN ≥ 1.

JCN is an INTEGER array of length LJCN. If ICNTL(2) = 0 or 1, JCN(K) must be set by the user to the column index of the entry whose row index is held in IRN(K), K = 1, 2, ..., NE. On exit, the contents of this array will have been altered. If ICNTL(2) = 2, the array is not accessed.

LA is an INTEGER variable that must be set by the user to the length of the array A. This argument is unchanged by the routine. **Restrictions:** If ICNTL(3) = 0, LA ≥ NE; if ICNTL(3) ≠ 0, LA ≥ 1.

A is a REAL (DOUBLE PRECISION in the D version, INTEGER in the I version, COMPLEX in the C version, or COMPLEX*16 in the Z version) array of length LA. If ICNTL(3) = 0, A(K) must be set by the user to the value of the entry whose row index is held in IRN(K), K = 1, 2, ..., NE. On exit, the array will have been permuted in the same way as the array IRN and, if the matrix is Hermitian (ICNTL(6) = ±2), any entries supplied by the user

lying in the upper triangular part of the matrix, will have been replaced by their complex conjugate. If ICNTL(3)≠0, the array is not accessed.

LIP is an INTEGER variable that must be set by the user to the length of the array IP. This argument is unchanged by the routine. **Restrictions:** If ICNTL(2)=0 or 2, LIP≥NC+1; if ICNTL(2)=1, LIP≥MAX(NC,NR)+1.

IP is an INTEGER array of length LIP. If ICNTL(2)=2, IP(J) must be set by the user to the position in the array IRN of the first entry in column J, J = 1, 2, ..., NC, and IP(NC+1) must be one greater than the number of entries in the matrix. In all cases, the array IP will have this meaning on exit from the subroutine.

LIW is an INTEGER variable which defines the length of the array IW. This argument is unchanged by the routine. **Restrictions:** LIW≥MAX(NC,NR)+1.

IW is an INTEGER array of length LIW. This array is used by the routine as workspace.

INFO is an INTEGER array of length 10 that need not be set on entry. On exit, a negative value of INFO(1) is used to signal a fatal error in the input data, a positive value of INFO(1) is used to indicate a warning, and a zero value is used to indicate a successful call to the routine. In cases of error, further information is held in INFO(2). Warnings are only given if ICNTL(1)=0 and, in this case, further information is provided in INFO(3) to INFO(7). INFO(8) to INFO(10) are not currently used by the routine and are set to zero. Possible nonzero values of INFO(1) and their meanings are as follows:

- -1 The restriction ICNTL(2) = 0, 1, or 2 has been violated. Immediate return with input parameters unchanged. Value of ICNTL(2) is given by INFO(2).
- -2 NC≤0. Immediate return with input parameters unchanged. Value of NC is given by INFO(2).
- -3 Error in NR. Either NR≤0 or ICNTL(6)=±1 or ±2 and NR≠NC. Immediate return with input parameters unchanged. Value of NR is given by INFO(2).
- -4 NE≤0. Immediate return with input parameters unchanged. Value of NE is given by INFO(2).
- -5 The parameter LJCN is too small. Immediate return with input parameters unchanged. Minimum value for LJCN is given by INFO(2).
- -6 The parameter LA is too small. Immediate return with input parameters unchanged. Minimum value for LA is given by INFO(2).
- -7 The parameter LIP is too small. Immediate return with input parameters unchanged. Minimum value for LIP is given by INFO(2).
- -8 The parameter LIW is too small. Immediate return with input parameters unchanged. Minimum value for LIW is given by INFO(2).
- -9 The entries of IP are not monotonic increasing (ICNTL(2)=2 only). The lowest column index for which monotonicity is not satisfied is returned in INFO(2).
- -10 For each I, either IRN(I) or JCN(I) is out of range (1≤I≤NE). The total number of indices in IRN and JCN that are out of range is returned in INFO(2).
- -11 The restriction ICNTL(6) = 0, ±1, or ±2 has been violated. Immediate return with input parameters unchanged. Value of ICNTL(6) is given by INFO(2).

- +1 One or more duplicated indices have been input. One copy of each such index is kept. If ICNTL(3)=0 and ICNTL(6)=1 or 2, the values of the duplicated entries are added together. If ICNTL(3)=0 and ICNTL(6)=-1 or -2, the value of the first occurrence of the entry is used.
- +2 One or more of the indices in IRN are out of range. These entries are removed by the routine.
- +4 One or more of the indices in JCN are out of range (ICNTL(2)=0 or 1). These entries are removed by the routine.

Positive values of INFO(1) are summed so that the user can identify all warnings issued by MC59, e.g. INFO(1)=+3 indicates both warnings +1 and +2 have been issued.

Statistics held in INFO(3) to INFO(7) are as follows:

INFO(3) Number of duplicate indices.

INFO(4) Number of out-of-range indices in IRN.

INFO(5) Number of out-of-range indices in JCN.

INFO(6) Number of entries in the matrix after the removal of duplicated and out-of-range indices.

INFO(7) Number of out-of-range indices in IRN lying in the upper triangular part of the matrix (ICNTL(2) = 2 and ICNTL(6) = ±1 or ±2 only). Note that INFO(7) ≤ INFO(4).

3 GENERAL INFORMATION

Use of common: None.

Workspace: An integer array IW of length LIW is used by MC59A/AD/AI/AC/AZ as workspace; see §2.1.

Other routines called directly: If ICNTL(2) = 0, the routine MC59A/AD/AI/AC/AZ calls the internal subroutines MC59B/BD/BI/BC/BZ and MC59E/ED/EI/EC/EZ. If ICNTL(2) = 1, the internal subroutines MC59B/BD/BI/BC/BZ, MC59C/CD/CI/CC/CZ, and MC59E/ED/EI/EC/EZ are called. If ICNTL(2) = 2, the internal subroutines MC59D/DD/DI/DC/DZ and MC59F/FD/FI/FC/FZ are called.

Input/output: Error messages on unit ICNTL(4) and warning messages on unit ICNTL(5). Printing of error or warning messages is suppressed by setting ICNTL(4) or ICNTL(5) to less than 0.

Restrictions: The routine MC59A/AD/AI/AC/AZ has the following restrictions:

NC ≥ 1, NR ≥ 1, NE ≥ 1,
if ICNTL(6) = ±1 or ±2, NR = NC
ICNTL(2) = 0, 1, or 2.
ICNTL(6) = 0, ±1, or ±2.
if ICNTL(2) = 0, LJCN ≥ NE, and LIP ≥ NC+1,
if ICNTL(2) = 1, LJCN ≥ NE, and LIP ≥ MAX(NC, NR)+1,
if ICNTL(2) = 2, LJCN ≥ 1, and LIP ≥ NC+1,
if ICNTL(3) = 0, LA ≥ NE,
if ICNTL(3) ≠ 0, LA ≥ 1,
LIW ≥ MAX(NC, NR)+1.

4 METHOD

MC59A/AD first checks the scalar input data for errors. A negative flag is set and control is immediately returned to the calling program if an error is found.

To sort the entries into column order, with arbitrary order within each column, the following procedure is used. The array JCN is scanned to count the number of entries in each column. If the user has chosen to check the matrix input data, a check is made for indices in JCN and IRN that are out of range. Any such entries are removed and a positive flag is set. The array IP is then set to hold the positions of the leading entries of the columns in the sorted form and these are copied to IW. The variable l is initialised to one. At an intermediate stage in the sort, the row indices of the first l–1 columns will be in their new positions. For $j = l$,..., NC, positions IP(j) to IW(j)–1 of IRN will hold row indices for column j. The entry currently stored in position IW(l) of IRN is now taken to be the current entry and examined to see if it is in place. If not, it is put into position IW(j), the value of IW(j) is increased by one, and the displaced entry is made the current entry. This chain of displacing entries continues until an entry that belongs to column l is found. This entry is stored in the position vacated by the first entry in the chain and IW(l) is increased by one. The next item in column l is then examined, unless column l has now been fully sorted, in which case column l+1 is considered. If the user requires the numerical values of the entries of the matrix to be sorted by columns then, at each step of the sort, the array A, which holds the numerical values, is permuted in the same way as the array IRN.

Once the entries have been sorted into column order, if the user has chosen to include matrix error checking, a

check is made for duplicates. The array IW is set to zero and each column is then checked in turn. If an entry in column J has row index I, IW(I) is set to J, unless IW(I) is already equal to J, in which case we have a duplicate. If duplicate entries are found, a positive flag is set and one copy of each duplicated entry is retained. If the numerical values of the entries of the matrix are being sorted, the numerical values of the duplicate entries are summed unless ICNTL(6) = -1 or -2. In this case, the value of the first occurrence of the entry in the sorted matrix is retained.

To sort the entries from arbitrary order to column order, with ordering by increasing row indices within each column, the above procedure is followed using rows in place of columns to obtain an ordering by rows, with arbitrary order within each row. At this stage, the entries are optionally checked for duplicates and out-of-range entries. The number of entries in each column is then obtained by a counting pass. The position of each entry is determined by examining each row in turn using a similar algorithm to that discussed above. The entries are finally permuted into these positions.

To sort the entries from column order with arbitrary order within each column to column order with ordering by increasing row indices within each column, a pairwise interchange algorithm is used of maximum order $r(r-1)/2$, for each column, where r is the number of entries in the column.

5 EXAMPLE OF USE

As a simple example, suppose we wish to order the sparse matrix

$$\begin{pmatrix} 1.1 & 1.2 & & 1.4 \\ & 2.2 & & \\ 3.1 & & 3.3 & 3.4 \\ & & & 4.4 \\ & 5.2 & & \end{pmatrix}$$

by columns, with the entries ordered by increasing row indices within each column. Using the following program:

```
      INTEGER          MAXN, MAXNZ
      PARAMETER       (MAXN=10, MAXNZ=30)
      INTEGER          NC,NR,NE,LA,LIP,LIW,LJCN,J,K,K1,K2
      INTEGER          IRN(MAXNZ), JCN(MAXNZ), IW(MAXN+1), IP(MAXN+1)
      DOUBLE PRECISION A(MAXNZ)
      INTEGER          ICNTL(10), INFO(10)
      EXTERNAL         MC59AD

C Initialise ICNTL.
C Require data checking.
      ICNTL(1) = 0
C Order output within each column.
      ICNTL(2) = 1
C Matrix entries, not just pattern.
      ICNTL(3) = 0
C Errors/warnings output on unit 6.
      ICNTL(4) = 6
      ICNTL(5) = 6
C Matrix unsymmetric.
      ICNTL(6) = 0
C Read matrix data.
      READ(5,*) NR,NC,NE
      DO 10 K = 1,NE
         READ(5,*) IRN(K), JCN(K), A(K)
   10 CONTINUE
C Set array lengths for MC59.
      LJCN = NE
      LA = NE
      LIP = MAX(NC,NR) + 1
```

```
      LIW = NR + 1
      CALL MC59AD(ICNTL,NC,NR,NE,IRN,LJCN,JCN,LA,A,LIP,IP,LIW,IW,INFO)
C Check error flag.
      WRITE(6,9000) INFO(1)
      IF (INFO(1).LT.0) STOP
C Write out ordered matrix.
      DO 30 J = 1,NC
         K1 = IP(J)
         K2 = IP(J+1)-1
         IF(K1 .LE. K2) THEN
            WRITE(6,9010) J
            DO 20 K = K1,K2
               WRITE(6,9020) IRN(K), A(K)
   20       CONTINUE
         END IF
   30 CONTINUE

 9000 FORMAT( 'INFO(1) = ',I2, ' on exit from MC59AD ')
 9010 FORMAT( 'Column ', I2)
 9020 FORMAT( 'Row ', I2, ' Value ', F4.1)

      STOP
      END
```

on the data

```
    5   4   9
    1   1   1.1
    3   3   3.3
    3   4   3.4
    3   1   3.1
    1   2   1.2
    1   4   1.4
    2   2   2.2
    4   4   4.4
    5   2   5.2
```

produces the output:

```
INFO(1) =  0 on exit from MC59AD
Column  1
Row  1 Value  1.1
Row  3 Value  3.1
Column  2
Row  1 Value  1.2
Row  2 Value  2.2
Row  5 Value  5.2
Column  3
Row  3 Value  3.3
Column  4
Row  1 Value  1.4
Row  3 Value  3.4
Row  4 Value  4.4
```